Microprocessors and Microsystems 32 (2008) 386–393

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A tunable high-performance architecture for enhancement of stream video captured under non-uniform lighting conditions

Ming Z. Zhang, Ming-Jung Seow, Li Tao, Vijayan K. Asari *

Computational Intelligence and Machine Vision Laboratory, Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA

ARTICLE INFO

Article history:
Available online 4 May 2008

Keywords:
Nonlinear color image enhancement
Reflectance-illuminance model
HSV-domain components rendering
Color space conversion
Log-domain computation
2D convolution
Parallel-pipelining
Multiplier-less architecture
Homomorphic filter
Quadrant symmetric architecture

ABSTRACT

A novel architecture for performing hue-saturation-value (HSV) domain enhancement of digital color images captured under non-uniform lighting conditions is proposed in this paper for video streaming applications. The approach promotes log-domain computation to eliminate all multiplications, divisions and exponentiations utilizing the compact high-speed logarithmic estimation modules. An optimized quadrant symmetric architecture is incorporated into the design of homomorphic filter for the enhancement of intensity value. Efficient modules are also presented for conversion between RGB and HSV color spaces with tunable H and S components in HSV for more flexible color rendering. The design is able to bring out details hidden in shadow regions of the image and preserve the bright parts with adjustable vividness and color shift for improvement of visual quality while maintaining its consistency. It is capable of producing 187.86 million outputs per second (MOPs) on Xilinx's Virtex II XC2V2000-4ff896 field programmable gate array (FPGA) at a clock frequency of 187.86 MHz. It can process over 179.1 (1024×1024) frames per second, which is very suitable for high definition videos, and consumes approximately 70.7% and 76.8% less hardware resource with 127% and 280% performance boost when compared to the designs with machine learning algorithm in [M.Z. Zhang, M.J. Seow, V.K. Asari, A high performance architecture for color image enhancement using a machine learning approach, International Journal of Computational Intelligence Research – Special Issue on Advances in Neural Networks 2(1) (2006) 40–47], and with separated dynamic and contrast enhancements in [H.T. Ngo, M.Z. Zhang, L. Tao, V.K. Asari, Design of a high performance architecture for real-time enhancement of video stream captured in extremely low lighting environment, International Journal of Embedded Systems: Special Issue on Media and Stream Processing, in press], respectively. This approach also provide 83.4 times performance gain with more consistent fidelity in the results compared to some DSP based implementations (256×256 frame size) [G.D. Hines, Z. Rahman, D.J. Jobson, G.A. Woodell, DSP implementation of the retinex image enhancement algorithm, visual information processing XIII, in: Proceedings of the SPIE, vol. 5438, 2004, pp. 13–24; G.D. Hines, Z. Rahman, D.J. Jobson, G.A. Woodell, Single-scale retinex using digital signal processors, in: Proceedings of the Global Signal Processing Conference, September 2004, pp. 1–6] under the reflectance-illuminance category of image enhancement models.

1. Introduction

Physical limitations exist in the sensor arrays of imaging devices, such as CCD and CMOS cameras. Often, the videos captured by these devices cannot represent scenes well that have both very bright and dark regions. The sensor cells are commonly compensated with the amount of saturation from bright regions, fading out the details in the darker regions. Enhancement algorithms [1–4] provide good rendering to bring out the details hidden due to dynamic range compression of the physical sensing devices. For applications in color images these algorithms may fail to preserve the color relationship among RGB channels which result in distortion of color information after enhancement. Thus, there is still room for the improvements. The recent development of fast converging neural network based learning algorithm called ratio rule [5,6] provides excellent solution for natural color restoration of the image after gray-level image enhancement. Hardware implementation of such algorithms is absolutely essential to parallelize the computation and deliver real-time throughputs for color images or video processing containing extensive transformations and large volumes of pixels. Implementation of window related operations such as convolution, summation, and matrix dot products which are common in enhancement architectures demands enormous amount of hardware resources [7,8]. Often, large

* Corresponding author. Tel.: +1 757 683 3752.
E-mail addresses: mzhan002@odu.edu (M.Z. Zhang), mseow@odu.edu (M.-J. Seow), ltaox001@odu.edu (L. Tao), vasari@odu.edu (V.K. Asari).

0141-9331/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2008.04.002

number of multiplications/divisions is needed [9]. Some designs compromise this issue by effectively adapting the architectures to very specific forms [7,8,10] and cannot operate on different sets of properties related to the operation without the aid of reconfiguration in FPGA based environment. We proposed the concept of log-domain computation in [11] to solve the problem of multiplication and division in the enhancement system and significantly reduce the hardware requirement while providing high throughput rate.

Algorithms developed under the reflectance-illuminance category of the image processing models are not unique. The origin for the theorization of such model for visual representation dates back to early 1970's [12] with stochastic image processing in [13] to reduce the salt-and-pepper noise (imposed from poor quality sensing device available at the time being). In classical approaches, homomorphic processing operates exclusively on the grayscale images. Recently, the concept has become popular for adapting the model to color image representation. Although the concepts for a number of exotic approaches are generalized by Kimmel et al. [14], dedicated architectures for such algorithms are generally unavailable to our best knowledge; thus, comparison is limited to existing designs relevant to the subject. One of the few extensively explored and well adapted techniques (in both hardware and software) in this category is led by Jobson's research team with Multi-Scale Retinex (MSR) model [2,3,15]. By the nature of the algorithmic procedure, MSR is suitable for DSP based implementation discussed in [16] where the fast Fourier transform (FFT) and inverse FFT (IFFT) may be plugged in from DSP library [17,18]. Further improvement on MSR can be made for better color consistency to minimize influence from background color. Within the same category, we presented a hardware-efficient architecture in [19] for enhancement of the digital color images with non-uniform darkness using a Ratio learning algorithm [5,6] for color distortion correction. We also presented the nonlinear enhancement architecture in [20] based on [21] which results with similar quality on the output images. In this paper, we propose an alternative design of the system to significantly reduce hardware requirement while achieving similar fidelity in the enhanced images. The new architecture is capable of improving the brightness, contrast, and visual color quality simultaneously. It processes the images and streaming video in HSV-domain with the homomorphic filter (Homomorphic model is a developed concept in computer science field for grayscale image processing) and converts the results back to RGB representation with standard conversion factor formulated in [22].

The paper is organized as follows. A brief on the related design techniques is given in Section 2 to introduce the methodology employed for the implementation. The formulation of homomorphic processing is discussed in Section 3 along with comparison to other algorithms. The implementation is described in detail in Section 4 with hardware simulation and error analysis explained in Section 5. The efficiency and performance are presented in Section 6 with appropriate comparison to other architectures. We wrap up the paper with conclusion that the proposed work is more hardware efficient and provides higher throughput with similar fidelity in the results.

2. Related works

2.1 Quadrant symmetry property

As introduced in previous section, window based operations are very common in video processing technology such as generalized 2D convolution defined by (1) where the center pixel of the kernel $W(j_1, j_2)$ is overlapped with center pixel of the input image $I(m, n)$ under the window of consideration to produce the output image $O(m, n)$. Often, coefficients associated with these kernels are non-arbitrary and exhibit interesting properties. It is a waste of the computational power and resource allocation from hardware designers' perspective to not take advantage of certain symmetries within the kernels. Such symmetries are very common in the design of digital filters. In particular, we utilize the quadrant symmetry (QS) of the kernels to support convolution operations (digital filtering) defined by (2) and (3) for odd and even kernels, respectively. This preprocessing ideally saves close to 75% of the multiplications in addition to the replacement of the hardware multipliers discussed in [23]. Such optimization results with the architecture that is neither too specific nor generic while focuses the essential computation to single quadrant. It maintains the flexibility of redefining the filter characteristics at run-time (soft upgrade) without recompiling and reconfiguring the architecture (hard upgrade) by external systems. Examples of the filters qualified for QS property include both separable [24] and non-separable kernels. QS also encapsulates circularly symmetric kernels such as Gaussians and Laplacian of Gaussians used for smoothing and edge detection, respectively.

$$O(m,n)=\sum_{j_1=0}^{J_1}\sum_{j_2=0}^{J_2}W(j_1,j_2)\cdot I\left(m+j_1-\frac{J_1}{2}+1,n+j_2-\frac{J_2}{2}+1\right) \quad (1)$$

$$O(m,n)=\sum_{j_1=0}^{\frac{J_1}{2}-1}\sum_{j_2=0}^{\frac{J_2}{2}-1}W(j_1,j_2)\cdot I\left(m\pm j_1+\frac{J_1}{2},n\pm j_2+\frac{J_2}{2}\right)+W\left(\frac{J_1}{2},\frac{J_2}{2}\right)\cdot I(m,n) \quad (2)$$

$$O(m,n)=\sum_{j_1=0}^{\frac{J_1-1}{2}}\sum_{j_2=0}^{\frac{J_2-1}{2}}W(j_1,j_2)\cdot\left[I\left(m+j_1-\frac{J_1}{2}+1,n+j_2-\frac{J_2}{2}+1\right)+I\left(m-j_1+\frac{J_1}{2},n+j_2-\frac{J_2}{2}+1\right)+I\left(m+j_1-\frac{J_1}{2}+1,n-j_2+\frac{J_2}{2}\right)+I\left(m-j_1+\frac{J_1}{2},n-j_2+\frac{J_2}{2}\right)\right] \quad (3)$$

2.2. Log-domain computation

Multiplications and divisions become additions and subtractions with logarithmic transformations defined by (4) which require significantly less computational power. A number to the power n becomes a matter of arithmetic shift operation achievable within single clock cycle for n equals to power of two, or multiplication operation for any finite n in general. Eq. (4) states that the $\log_2$ scale of V can be calculated by concatenating the index I_V of leading 1s in V with the fractions (remaining bits after I_Vth bit). The reversed process holds true as well, except the leading 1s and fractions, L_f, are shifted to the left by L_i (integer of L) bits as shown in (4). This definition is generalized to integer values as well as fraction numbers. For example, $\log_2(0000.0110)$ binary becomes -1.5 or $(-2+0.5)$ in decimal since the position of I_v is -2 (two places after decimal point) with fraction 0.10 in binary. The correct value should be -1.415 which results with 6% error from the approximation for worst case scenario. Application of this concept eliminates most costly components just described for hardware designs. Thus, it is crucial to implement efficient logarithmic estimation modules in such a way that is very compact in its design, reduces large amount of hardware resource, and provides very high throughput rate [11,19]. Designs based on the concept presented in [25] which employees unrolled pipeline architectures such as [20,26,27] may not be efficient for replacement of

multiplications and divisions in window related architectures for FPGA based implementation. Particularly in filters, such architectures usually require large number of multiplications and the amount of hardware resource allocated for the unrolled pipeline stages usually can come close to the cost of the multipliers on FPGA technology. Our implementation of the estimation modules packs the resolution-dependent unrolled pipeline style design into a few stages regardless its resolution at the same time optimizes the component count, power and speed. It is about 10 times reduction in the resource and 170% performance boost in FPGA environment. We generalized the modules to support both integer and fraction numbers without introducing hardware complexity. These modules are also insensitive to the bit-resolution that exists in hardware multipliers in which the performance is inversely proportional to the number of bits in the multipliers. We have demonstrated the use of log-domain computations in [19,23] for image processing applications with a figure of 60% hardware reduction in addition to the applicable QS based architecture. The image enhancement model is discussed in the following section.

$$\log_2(V) \cong \{I_V\} + \{(V - I_V) \gg I_V\} \Longleftrightarrow$$
$$\log_2^{-1}(L) \cong \{1 \ll L_i\} + \{L_f \ll L_i\} \tag{4}$$

3. Reflectance-illuminance model

3.1. Homomorphic based HSV-domain enhancement

Color distortion correction can be avoided for color image enhancement in HSV-domain where the color (H), intensity (V) and saturation/color density (S) components can be rendered separately without introducing the distortion. To remove the shadows in the image, only the V component in HSV needs to be enhanced instead of boosting separate RGB channels which results with loss of color consistency. Extraction of the V component is defined by

$$V(x,y) = \max(R_{I(x,y)}, G_{I(x,y)}, B_{I(x,y)}), \tag{5}$$

where the $I(x, y)$ is the input image. The V component is enhanced by a homomorphic filter defined as

$$V_{\text{enh}}(x,y) = 2^{\left(\log_2\left(\frac{V(x,y)}{2^P}\right) * h(x,y)\right)} \times D \tag{6}$$

for $\log_2$ expression where the * denotes convolution operation, $h(x, y)$ is the $K \times K$ spatial-domain filter coefficients sampled from high-boosting homomorphic transfer function, P is the resolution of the pixel, D is the de-normalizing factor, and $V_{\text{enh}}(x, y)$ is the enhanced intensity value of the image. This enhancement model assumes that the detail (reflectance) and illuminance in the image are logarithmically separable [3,13,28]. Hence the model belongs to reflectance-illuminance category. The quadrant symmetry (QS) property of the homomorphic filter allows us to reduce the number of multiplications to about 25% as presented in [23]. The enhanced image can now be transformed back to RGB representation by mapping the following set according to i:

$$\{R'G'B'\}_n = \{\{e,p,t\},\{n,e,t\},\{t,e,p\},\{t,n,e\},\{p,t,e\},\{e,t,n\}\} \quad \text{for } i \text{ in}\{\{0\},\ldots,\{5\}\}, \tag{7}$$

where $t = 1 - S$, $n = 1 - S \times f$, $p = 1 - S \times (1 - f)$, $e = 1$, and $\{R'G'B'\}_n$ are the normalized enhanced RGB components. The i and f are the integer and fraction portions of H which is the angular representation of color component in HSV-domain defined by (8) where $L = \min(\text{RGB})$ and H_{os} is the color shifting factor. The S component in HSV-domain is defined to be (9) with color saturation/density rendering factor S_{os}. The final output, $\{R'G'B'\}$, can be calculated as (10) since the denominator is approximately one for non-uniform scenes or images which contain bright parts, where $V_{\text{enh}} = 2^{V_{\text{enhl}}} \times D$. Eqs. (5)–(10) provide basic framework for the design of HSV-domain enhancement system. The H component can be added with offset for color shifting effect while tuning the S component controls the vividness in visual appearance of the image.

$$H = \begin{cases} 0 + H_{\text{os}} + (G - B)/(V - L), & \text{if } V = R \\ 2 + H_{\text{os}} + (B - R)/(V - L), & \text{if } V = G \\ 4 + H_{\text{os}} + (R - G)/(V - L), & \text{if } V = B \end{cases} \tag{8}$$

$$S = (V - L) \times S_{\text{os}}/V \tag{9}$$

$$\{R'G'B'\} = \frac{\{R'G'B'\}_n \times V_{\text{enh}}}{\max(\{R'G'B'\}_n)} \text{ or } \{R'G'B'\}_n \times V_{\text{enh}} \tag{10}$$

3.2. Brief comparison of algorithms under the model

While discussion of the other enhancement algorithms is outside the scope of this paper, it is important to illustrate the results since we will compare the hardware utilization and the performance for the available implementation of the algorithms. The original test image is shown in Fig. 1a. After enhancing the image on separate RGB channels, more details are revealed as shown in Fig. 1b; however, the image appears pale due to lost of the color relationship between the channels. The result of enhancement by Multi-Scale Retinex with Color Restoration (MSRCR) [15], which is based on human perception, is illustrated in Fig. 1c. This approach corrects the color distortion but still appear grayish in certain areas depending on the background color and lighting condition. In this case the background has mild influence on the image. Thus further improvement can be made. The hardware implementation of this algorithm can be done but the large scale kernel of the filters makes it impractical to achieve in time domain. Shown in Fig. 1d is the output of the Luminance Dependent Nonlinear Image Enhancement (LDNE) algorithm presented in [21] which we imple-

Fig. 1. Algorithm comparison: (a) original image taken from [15], (b) enhanced RGB channels, (c) MSR with color correction [15], (d) LDNE [20] [21], (e) RR [5] [6] [19], and (f) enhanced with the approach we propose.

mented the hardware system in [20]. It is clear that the color is consistent which is obvious on the color of the hair of the man shown in the figure. Fig. 1e is the output of correcting Fig. 1b with ratio rule which is a machine learning algorithm [5,6]. We also implemented it in [19]. The output for the design we propose in this paper is illustrated in Fig. 1f. It has similar characteristics with Fig. 1d and e and is somewhere between the two. With carefully chosen homomorphic transfer function, it can be hard to distinguish by human eyes. Nonetheless, the difference between these designs in terms of the performance and hardware utilization is quite dramatic. Design of the proposed system architecture is discussed in next section.

4. Design of system architecture

4.1. Overview

A brief overview of the enhancement system is shown in Fig. 2 to illustrate basic computational sequence. The pixel values in RGB components are first sent to HSV homomorphic filter and RGB2HSV conversion block simultaneously. The *V* component is then extracted and enhanced by the high-boosting homomorphic filter while the H and S components are rendered. Lastly, the enhanced *H* and *S* components are combined with the enhanced *V* component in HSV2RGB module where the conversion back to RGB representation takes place. The architecture is capable of enhancing the stream video at run-time without frame buffering. Even though frame buffering is preferred, it is only necessary when the architecture cannot keep up with the throughput from the input source or the output cannot be synchronized with the displaying device.

4.2. Architecture of homomorphic filter unit

The homomorphic filter unit (HFU) coupled with an array of the $\log_2$ scaled *V* component, which is extracted by max filter architecture, is illustrated in Fig. 3. The QS property of the 2D convolution operation allows the computation to concentrate on one quarter of the kernel through folding, in addition to eliminating the multipliers [23]. The vertical folding of data is accomplished by linearly folding the $\log_2$ scaled data with adders from the buffers which create internal parallelism for massive concurrent processing. This halves the processing bandwidth. To normalize a value v ($\log_2(v/2^P) = \log_2(v) - P$), which is negative, given the fact that image pixels are unsigned and $\log_2$ of negative number is undefined, the absolute value can be logically approximated by taking the inverted output ($\overline{Q} \approx P - log_2(v) = \overline{log_2(v)}$) of the registered result from vertical folding. To reduce the processing bandwidth by another half, the horizontal folding defined by (2) and (3) is translated to (11) and (12) and performed with respect to even and odd dimension kernels, taking account of the delay in systolic architecture. The *H*-fold denotes the results from horizontal folding and HQ is a set of horizontal shift registers for vertically folded data. The registered results of the *H*-fold stage are sent to arrays of processing elements (PEs) for successive filtering. The partial results from the PE arrays (PEAs) are combined together by a pipelined adder tree (PAT). The overall output, V_{enhl}, is kept in $\log_2$ scale for the multiplications in HSV to RGB color space conversion.

$$H\text{-fold}(k) = \begin{cases} \mathrm{HQ}[0] + \mathrm{HQ}[2k+1], & \text{for odd } J_1,\ k \neq 0 \\ \mathrm{HQ}[0], & \text{for odd } J_1,\ k = 0 \end{cases} \tag{11}$$

$$H\text{-fold}(k) = \mathrm{HQ}[0] + \mathrm{HQ}[2k], \quad \text{for even } J_1, \forall k \tag{12}$$

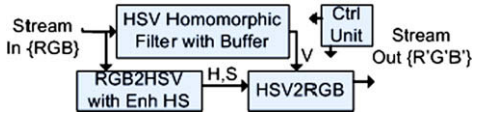

Fig. 2. Block diagram of the enhancement architecture.

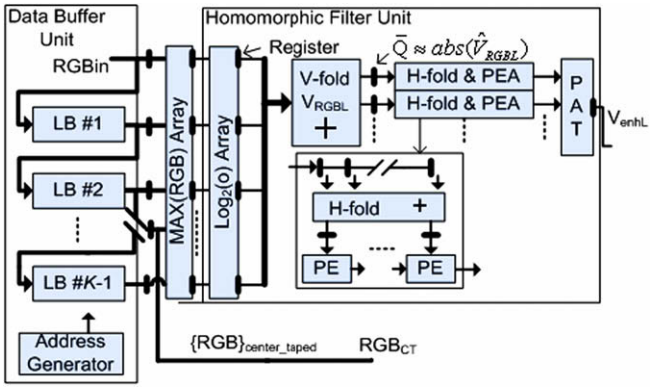

Fig. 3. Architecture of the HFU with the output in $\log_2$ scale.

4.2.1. Architecture of pipelined processing elements in homomorphic filter

The design of the PE in the homomorphic filter utilizes the log-domain computation to eliminate the need of hardware multipliers. The data from *H*-fold register is pre-normalized without extra logics by shifting the bus. It is then converted to $\log_2$ scale as shown in Fig. 4a and added with $\log_2$ scaled kernel coefficients (LKC) in LKC register set. The result from last stage is converted back to linear scale with range check (RC). If the overflow or underflow occurs, the holding register of this pipeline stage is set or clear, respectively. Setting and clearing contribute the max and min values representable to *N*-bit register. The output of this stage is de-normalized, likewise by bus shifting, before it is successively accumulated along the accumulation line. The $\log_2$ architecture shown in Fig. 4b is very similar to [23], except full precision is used and registers are introduced to approximately double the performance with two-stage pipeline. The maximum logic delay is reduced to single component (multiplexer) and makes no sense to pipeline beyond this point. The inverse-$\log_2$ is readily optimized with only one pipeline stage at its peak performance. So no modification is needed.

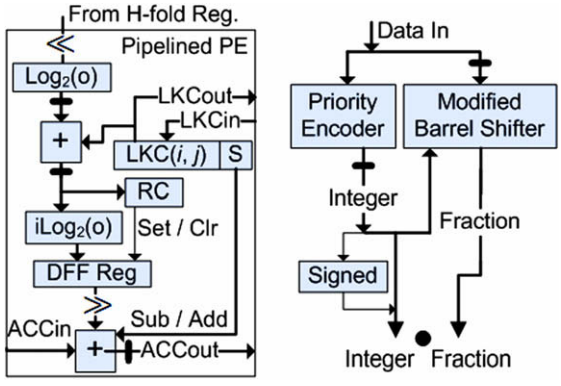

Fig. 4. Architecture of (a) the PE in the homomorphic filter, and (b) the pipelined $\log_2$ module.

4.3. Data buffer unit

The data buffer unit (DBU), which generates the internal parallelism for homomorphic filtering, is implemented with the dual port RAMs (DPRAMs) as shown in Fig. 5. One set of DPRAMs is utilized to form line buffers (LBs) and store just enough lines of image in the LBs to create massive internal parallelism for concurrent processing. The pixels are fetched into the DBU in raster-scan fashion. The DPRAM based implementation has advantage of significantly simplifying the address generator compared to commonly known first-in-first-out (FIFO) based approach. Tracking of items during transient stage is eliminated. Furthermore, only one address generator is necessary in DBU. It consists of two counters to automatically keep track of the memory locations to insert and read the data to internal PDB for extraction of V component. Data bus A (DBA) of $(K-1)\times P_{RGB}$ bits wide, which is formed with just enough number of DPRAMs in parallel, is used to insert pixel values through write-back paths to the memory location designated by address bus A (ABA). For 8-bit pixel resolution, P_{RGB} is 24 bits. The data bus B (DBB) is used to read the pixel values onto internal PDB and write to the write-back paths. The center tapped RGB (RGB_{CT}) components are sent to RGB2HSV color space conversion module.

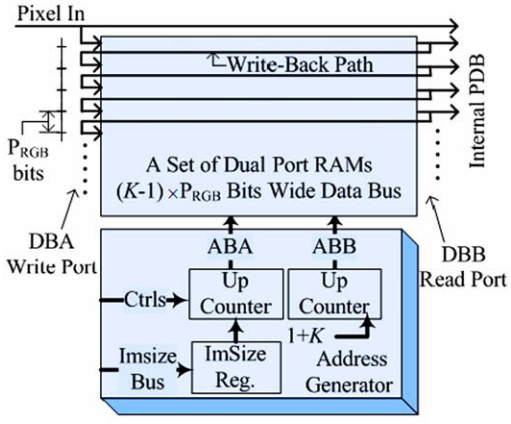

Fig. 5. Architecture of the DBU shown in Fig. 3. The data bus of $(K-1)$ PRGB bits wide is grouped into a number of 24-bit paths to form effective LBs for 8-bit pixel resolution.

4.4. Architecture for RGB2HSV color space conversion

The architecture for the conversion of RGB–HSV components is shown in Fig. 6. The calculation is performed in parallel with the HFU. To compute the essential components of (8) and (9), the subtractions between RGB channels are carried out in pipeline stage 1, p1. The sign bits are used to map the min/max RGB components in (5) and (8) to H_{min}/H_{max} with two bits respectively where the numerator of (8), V, and L components can be determined with multiplexers (MUX) on p2. The sign for the numerator of (8) is extracted for arithmetic calculation between the integer parts and fractions of (8) at later stage. The division of positive numbers can then be performed with subtraction in log-domain. The H_{max} is further wrapped (WrapZN) if the numerator of (8) results with negative fraction [22]. The $\log_2$ scaled V and $(V-L)$ components are calculated at p5 and subtraction of the two determines the S component in $\log_2$ scale (S_{nl}). The fraction of H component is calculated likewise. To perform rendering of the color density, the S_{nl} is added with S_{OS} to produce S_L which is used for multiplication in HSV to RGB conversion module. At the last stage of the pipeline, the fraction of H component is combined (according to the sign bit determined from earlier stage) with the decimal which is calculated with the WrapZN output (right shifted to reflect the integer portion of (8)) and the offset value (H_{OS}) for color shifting effect. The V component is discarded since the HFU produces the enhanced V. The architecture for converting HSV back to RGB space is discussed next.

4.5. Architecture for HSV2RGB color space conversion

Architecture for restoring RGB representation is illustrated in Fig. 7. The positive integer, H_{int}, of H component calculated in Section 4.4 is first modulated by 6 to prevent overflow and then used to select the n, p, t, and e components of (7) with MUXes. The fraction, H_{frac}, is converted to $\log_2$ scale to compute $S_n \times H_f$ with adder and converted to linear scale with $i\log_2$ module. The S_L component is also converted back to linear scale. With these partial components available, p is rearranged to $S_n \times H_f + (1 - S_n)$ to eliminate separate calculation as illustrated in Fig. 7 (2nd input on MUX). Once the $R'G'B'_n$ are selected by H_{int}, the sign bits are extracted since the results from rendering S component can be negative. The absolute values of $R'G'B'_n$ are converted to $\log_2$ scale to produce $r'g'b'$ with over/under-flow RC. In log-domain, the $r'g'b'$ channels are added or subtracted, depending on the signs determined by last

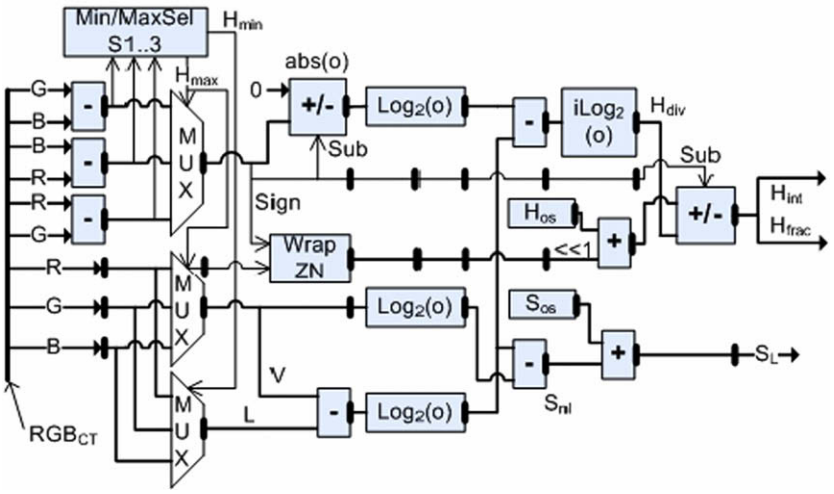

Fig. 6. Architecture of the RGB to HSV color space convertor.

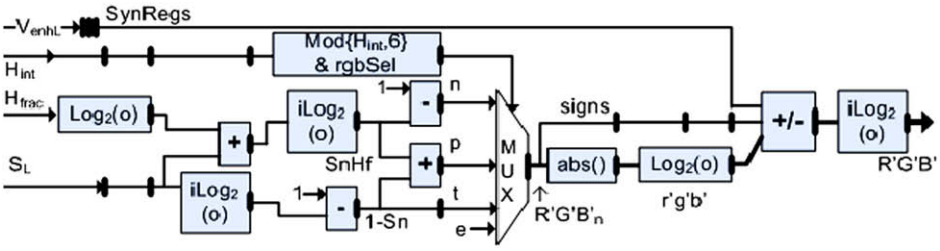

Fig. 7. Architecture of the HSV to RGB color space convertor.

pipeline stage, with the V_{enhl} (enhanced V synchronized to the module with SynRegs) computed by HFU. The final output is calculated by taking the results back to linear scale with RC. Thus, calculation of (10) is obtained. This completes the discussion on architectural design.

5. Hardware simulation and error analysis

The image is sent to the architecture pixel by pixel in raster-scan fashion which is common for video streaming. After the transient state, the output becomes available and is collected for error

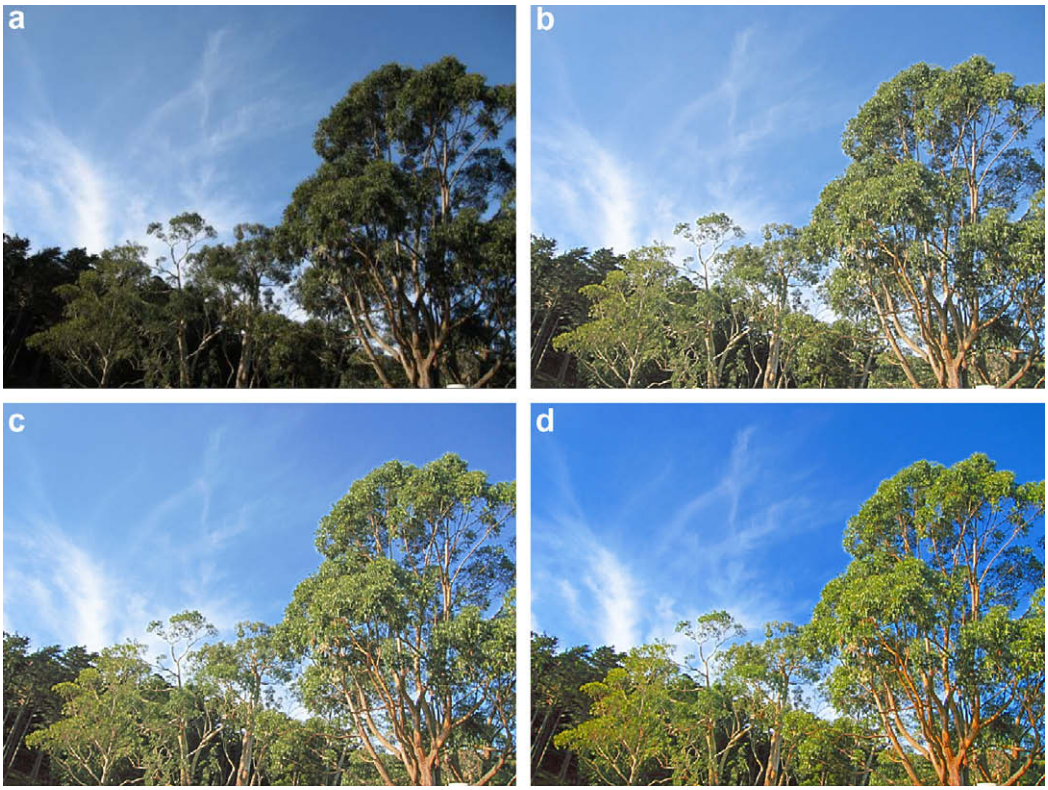

Fig. 8. Results from software and hardware simulations (b) and (c) with input image (a). The architectural simulation with S component rendering is illustrated in (d).

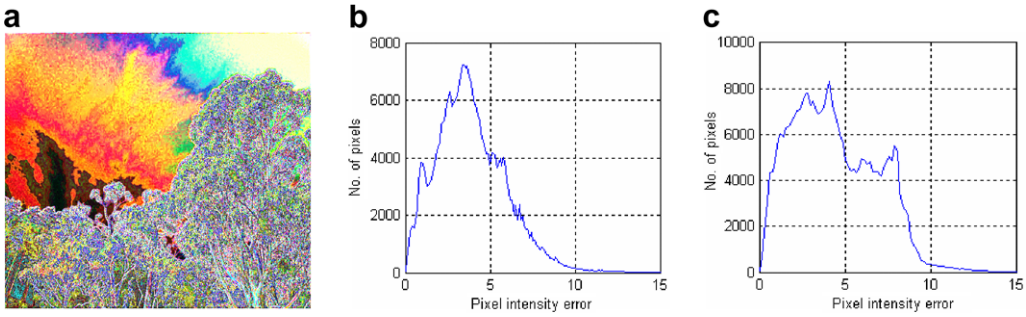

Fig. 9. Error characteristics. (a) Difference error scaled by 50 times. (b) Histogram of the error from (a). (c) Histogram of the error with Sos (1.5) rendering at its worst case scenario.

analysis. The overall output of the enhancement architecture is recorded to give graphic view of the enhanced image for quick evaluation of the visual quality. Typical test image is shown in Fig. 8a where the shadow regions exist as the consequence of the saturation in bright regions. The outputs of the system produced by software and hardware simulations without H_{os} and S_{os} rendering are illustrated in Fig. 8b and c, respectively. The result of architectural simulation with S_{os} set to 1.5 is illustrated in Fig 8d. As one can see that majority of the detail hidden in the dark regions are brought out while the natural color is preserved. Overall, the visual quality is very satisfied with fewer shadow regions.

The difference between the software and architectural simulations is shown in Fig. 9a scaled by 50 times. Typical histograms of the error with the data from Fig. 9a are illustrated in Fig. 9b. The average error of the system is 3.95 pixel intensities with respected to the test image. Simulation with large set of images shows majority of the errors in this system is below 10 with the average error around 4. The histogram for the test with S_{os} (1.5) parameter is shown in Fig. 9c. The average error increases to 4.51. The additional error came from S_{os} which is set to the value that is at one of the worst case scenarios [23]. The error measure of Fig. 9 includes the fact that the hardware simulation is bounded to approximation error and specific number of bits representable in the architecture. While the hardware simulation shows very attractive results, the efficiency of hardware utilization and its performance is also very important.

6. Resource utilization and performance evaluation

The hardware resource utilization is characterized based on the Xilinx's Virtex II XC2V2000-4ff896 FPGA on Xilinx's multimedia platform and the Integrated Software Environment (ISE) [29]. The particular FPGA chip we target has 10,752 logic slices, 21,504 flip-flops (FFs), 21,504 lookup tables (4-input LUTs), 56 block RAMs (BRAMs) and multipliers; however, we do not utilize the built-in multipliers. The resource allocation for various sizes of the kernels in homomorphic filter is shown in Table 1. For 9×9 filter kernel, the computational power is approximately 81 multipliers which is significantly less compared to [19] with similar settings where 243 multipliers and 150 dividers are needed if conventional approach is taken. With the alternative algorithm introduced in this design and the application of log-domain computation the amount of hardware resource necessary for this implementation is tremendously reduced. The maximum kernel size which can be utilized on target FPGA consumes 82% of the logic slices, 48% of the FFs, 48% of LUTs and 22 BRAMs. Testing conducted in Section 5 has shown that a 7×7 filter kernel is sufficient to remove most shadows of reasonable darkness. Only 21% of the logic slices is needed in this case. The proposed design uses approximately 70.7% and 76.8% less logic slices on average with great performance boost compared to the architectures presented in [19] and [20] (1024×1024 frame size), respectively.

The critical timing analysis of Xilinx's ISE shows that 187.86 MOPS is the most optimal throughput achievable with the maximum clock frequency of 187.86 MHz. Further evaluation of pipelining the critical path suggests that increasing the level of pipeline does not gain significant throughput rate. This directly indicates the impact of the design with tightly coupled and well pipelined system. Given 1024×1024 image frame, it can process over 179.1 frames per second at its peak performance without frame buffering, which is very suitable for video streaming applications. This tremendous gain in the performance while consuming significantly less hardware resources would have been extremely difficult to achieve without the algorithmic simplification, efficient filter design and log-domain computation. The additional benefit is that the filter coefficients are not hardwired, which gives the highest flexibility in reloading the coefficients without the need of dynamic reconfiguration for different characteristics of the transfer functions. The performance of the proposed approach increases to 127% and 280% when compared to the designs we presented in [19] and [20] (1024×1024 frame size), respectively. Due to massive parallelism, it is also far superior to those DSP based approaches discussed in [16,20] which utilize limited number of functional units. A comparison of the proposed work with other implementations most relevant to the model is listed in Table 2. While the throughput of the FPGA based architectures significantly out performs those of DSP processors, it should be pointed out that the DSP processors are largely constraint to the available functional units with associated resource. So the full level parallelism cannot be exploited. For this reason, the processors usually operate at higher clock frequency to achieve minimum real-time criteria with limited video resolution. The operating frequency to throughput ratio can be more than two orders of magnitudes (i.e. 100 clocks/pixel). For standard NTSC (720×480 at 30 fps) video with the algorithms fully exploited, the DSP processors need to operate at GHz scale where the memory access of the systems becomes the bottleneck without the assistance of improved memory management [30]. Despite the drawback, DSP based implementations are still well adopted for lower end applications where the perfor-

Table 1
Hardware resource utilization for various sizes of the kernels with corresponding throughput rate

Kernel size	Logic slices (%)	Slice FFs (%)	LUTs (%)	BRAMs	Perf (MOPS)
5×5	14	8	9	6	186.25
7×7	21	12	13	8	186.25
9×9	30	17	18	11	187.86
11×11	40	23	24	14	187.86
13×13	53	31	31	16	187.86
15×15	66	39	39	19	187.86
17×17	82	48	48	22	187.86

Table 2
Comparison of the proposed work with other implementations most relevant to reflectance-illuminace model. Note that 256×256 frame size (should be power of 2) is employed so the performance is not penalized for [16] and [30] to utilize FFT and IFFT

Hardware platforms	FPGA[a][20]: XC2V2000	FPGA[a][19]: XC2V2000	DSP[b][16]: C6711	DSP[b][16]: C6713	DSP[b][30]: DM642	FPGA[a,j]: XC2V2000
Operating frequency	67 MHz	147.3 MHz	150 MHz	225 MHz	600 MHz	186.3 MHz
Nature of design	Systolic-parallel	Systolic-parallel	VLIW (256-bit)	VLIW (256-bit)	VLIW (256-bit)	Systolic-parallel
Resource utilization	49.3% logic slices[h]	46% logic slices[i]	DSP + DSK[c] support	DSP + DSK[c] support	DSP + EVM[d] support	14% logic slices[i]
Frame buffers	Ext. 133 MHz ZBTRAMs	None	Ext. 100 MHz SDRAMs	Ext. 90 MHz SDRAMs	Ext. 133 MHz SDRAMs	None
Throughput rate	1ppc[e] 67mpps[f]	1ppc[e] 147mpps[f]	0.009ppc[e] 1.36mpps[f]	0.008ppc[e] 1.84mpps[f]	0.004ppc[e] 2.24mpps[f]	1ppc[e] 186mpps[f]
Frame rate	1022 fps[g]	2248 fps[g]	20.7 fps[g]	28 fps[g]	34.1 fps[g]	2843 fps[g]

Notes: [a]Xilinx's Virtex II XC2V2000-4ff896 FPGA on multimedia platform. [b]Texas Instruments' DSPs in TMS320 family with appropriate platforms ([c]DSK EVM[d]) and 2 levels cache support. [e]Ppc is unit for pixels per processor clock. [f]Mpps is unit for million pixels per second. [g]Fps is unit for frames per second. [h]Use of all embedded multipliers not included. [i]Architecture is multiplier-less. [j]Proposed work in this paper utilizes same technology as [19] and [20].

mance of the systems is not critical. The performance of [20] was limited to uneven pipelining and the setup of external Zero Bus Turnaround (ZBT) RAMs which are coupled with the core module. This bottleneck does not impose on the proposed architecture since the throughput is sufficient to enhance the video on the fly at the constant rate as the streamed video. Overall, the new design achieved similar output quality with reduced hardware resource while boosted the performance.

7 Conclusion

We presented a novel architecture for enhancement of digital color images captured under non-uniform lighting conditions for video processing applications. The design minimized the color distortion by enhancing the images in Hue-Saturation-Value color space. With very efficient generalized logarithmic estimation modules, the optimized quadrant symmetric architecture and low complexity color space converters, the design was able to utilize significantly less hardware resource while provided high throughput rate when compared to the approaches implemented with the ratio learning algorithm [19], and the nonlinear video stream enhancement model [20]. It was observed on Xilinx's ISE development environment with multimedia platform that the throughput of the proposed architecture is 127% and 280% higher with 70.7% and 76.8% reduction in hardware resource when compared to [19] and [20], respectively. To reduce the implication of technological dependency in the implementation, the comparison was done under same environment provided by Xilinx's development kit. We showed that the design was able to bring out details hidden in shadow regions of the image without saturating out the bright areas. Furthermore, it was able to produce 187.86 million outputs per second (MOPs) on Xilinx's Virtex II XC2V2000-4ff896 FPGA technology at a clock frequency of 187.86 MHz. For 1024×1024 images, it can process over 179.1 frames per second and consumes 30% of the logic slices for 9×9 taps of the homomorphic filter. The performance is at least 83.4 times greater compared to MSR architectures embedded in DSP environment [2,16,30]. Research is progressing to give broader optimization scheme between the required performance and the hardware resource constraints, driven by applications, without compromising the flexibility of the architecture.

References

[1] V. Caselles, J.L. Lisani, J.M. Morel, G. Sapiro, Shape preserving local histogram modification, IEEE Transaction on Image Processing 8 (2) (1999) 220–230.
[2] D.J. Jobson, Z. Rahman, G.A. Woodell, A multi-scale retinex for bridging the gap between color images and the human observation of scenes, IEEE Transaction on Image Processing 6 (7) (1997) 965–976.
[3] E.H. Land, J.J. McCann, Lightness and retinex theory, Journal of the Optical Society of America 61 (1) (1971) 1–11.
[4] R. Mekle, A.F. Laine, S.J. Smith, Evaluation of a multi-scale enhancement protocol for digital mammography, in: R.N. Strickland (Ed.), Image-Processing Techniques for Tumor Detection, Marcel Dekker, New York, NY, 2001, pp. 155–186.
[5] M.J. Seow, V.K. Asari, Ratio rule and homomorphic filter for enhancement of digital color image, Journal of Neurocomputation 69 (7–9) (2006) 954–958.
[6] M.J. Seow, V.K. Asari, Associative memory using ratio rule for multi-valued pattern association, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Portland, Oregon, 2003, pp. 2518–2522.
[7] A.F. Breitzman, Automatic derivation and implementation of fast convolution algorithms, PhD Dissertation, Drexel University, 2003.
[8] E. Jamro, Parameterised automated generation of convolvers implemented in FPGAs, PhD Dissertation, University of Mining and Metallurgy, 2001.
[9] A. Wong, A new scalable systolic array processor architecture for discrete convolution, MS thesis, University of Kentucky, 2003.
[10] J. Yli-kaakinen, T. Saramaki, A systematic algorithm for the design of multiplierless FIR filters, in: Proceedings of the IEEE International Symposium Circuits and Systems, vol. 2, Sydney, Australia, 2001, pp. 185–188.
[11] M.Z. Zhang, H.T. Ngo, K.V. Asari, Multiplier-less VLSI architecture for real-time computation of multi-dimensional convolution, Journal of Microprocessors and Microsystems 31 (2007) 25–37.
[12] T.G. Stockham Jr., Image processing in the context of a visual model, Proceedings of the IEEE 60 (1972) 828–842. July.
[13] R.W. Fries, J.W. Modestino, Image enhancement by stochastic homomorphic filtering, IEEE Transactions on Acoustics Speech and Signal Processing 27 (6) (1979) 37–625.
[14] R. Kimmel, M. Elad, D. Shaked, R. Keshet (Kresch), I. Sobel, A variational framework for retinex, The International Journal on Computer Vision 52 (1) (2003) 7–23. (April).
[15] Z. Rahman, D.J. Jobson, G.A. Woodell, Retinex processing for automatic image enhancement, Journal of Electron Imaging 13 (1) (2004) 1–12.
[16] G.D. , Springer, 2005.
[19] M.Z. Zhang, M.J. Seow, V.K. Asari, A high performance architecture for color image enhancement using a machine learning approach, International Journal of Computational Intelligence Research – Special Issue on Advances in Neural Networks 2 (1) (2006) 40–47.
[20] H.T Ngo, M.Z. Zhang, L. Tao, V.K. Asari, Design of a high performance architecture for real-time enhancement of video stream captured in extremely Low lighting environment, International Journal of Embedded Systems: Special Issue on Media and Stream Processing, in press.
[21] L. Tao, V.K. Asari, Modified luminance based MSR for fast and efficient image enhancement, IEEE International Workshop on Applied Imagery and Pattern Recognition, Washington, DC, USA, AIPR-2003, pp. 174–179.
[22] Color Space Conversions: <http://www.cs.rit.edu/~ncs/color/t_convert.html>.
[23] M.Z. Zhang, K.V. Asari, An efficient multiplier-less architecture for 2D convolution with quadrant symmetric kernels, Integration, the VLSI Journal 40 (4) (2007) 490–502.
[24] V. Lakshmanan, A separable filter for directional smoothing, IEEE Transactions on Geoscience and Remote Sensing Letters 1 (3) (2004) 192–195.
[25] J.N. Mitchell, Computer multiplication and division using binary logarithms, IRE Transactions on Electronic Computers (1962) 512–517.
[26] K.H. Abed, R. Siferd, CMOS VLSI implementation of 16-bit logarithm and anti-logarithm converters, in: Proceedings of the IEEE Midwest Symposium on Circuits and Systems, August 2000, pp. 776–779.
[27] D.J. Mclaren, Improved Mitchell-based logarithmic multiplier for low-power DSP applications, SOC Conference, 2003, in: Proceedings of the IEEE International [Systems-on-Chip], September 2003, pp. 53–56, 17–20.
[28] L. Tao, K.V. Asari, An efficient illuminance-reflectance nonlinear video stream enhancement model, in: Proceedings of the IS&T/SPIE Symposium on Electronic Imaging: Real-Time Image Processing III, vol. 6063, San Jose, CA, pp. 60630I-1-12, 2006.
[29] Xilinx's FPGA board: http://www.xilinx.com/bvdocs/userguides/ug020.pdf.
[30] G.D. Hines, Z. Rahman, D.J. Jobson, G.A. Woodell, Single-scale retinex using digital signal processors, in: Proceedings of the Global Signal Processing Conference, September 2004, pp. 1–6.